

Cohort of LSTM and lexicon verification for handwriting recognition with gigantic lexicon

Bruno STUNER^{a,*}, Clément CHATELAIN^a, Thierry PAQUET^a

^a*Normandie Univ, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS, 76000 Rouen, France*

Abstract

Handwriting recognition state of the art methods are based on Long Short Term Memory (LSTM) recurrent neural networks (RNN) coupled with the use of linguistic knowledge. LSTM RNN presents high raw performance and interesting training properties that allow us to break with the standard method at the state of the art. We present a simple and efficient way to extract from a single training a large number of complementary LSTM RNN, called cohort, combined in a cascade architecture with a lexical verification. This process does not require fine tuning, making it easy to use. Our verification allow to deal quickly and efficiently with gigantic lexicon (over 3 million words). We achieve state of the art results for isolated word recognition with very large lexicon and present novel results for an unprecedented gigantic lexicon.

Keywords: Handwriting Recognition; Recurrent Neural Network; LSTM; Very Large Vocabulary; Gigantic Lexicon; Cascade of LSTM; Lexicon Verification; Cohort

1. Introduction

Handwriting recognition is the numeric process of translating handwritten text images into strings of characters. The handwriting recognition process

*Corresponding author

Email addresses: bruno.stuner@litislab.eu (Bruno STUNER), clement.chatelain@litislab.eu (Clément CHATELAIN), thierry.paquet@litislab.eu (Thierry PAQUET)

traditionally involves two steps [1] optical character recognition and linguistic processing. Optical character recognition is a hard task due to the variability of shapes in handwritten texts, since every human has his own personal writing style. Therefore, even when using state of the art classifiers like deep neural networks to recognize characters [2], a considerable amount of errors would occur by considering only the optical model. Linguistic processing aims at combining the characters hypothesis together so as to provide the most likely sequence of words in accordance with some high level linguistic rules. There are two types of linguistic knowledge: lexicons and language models which are probabilistic modelization of the language. The use of linguistic knowledge is an open problem. Which lexicon resource should be used, which corpora should be selected for training the language model ? These choices directly affect the recognition performance. In the case of lexicon driven methods, where the characters are aligned on the lexicon words, too small lexicons fail to cover the test data set, thus missing solutions. However using a large lexicon (1000 words and more), would require many computations and sometimes generate precision loss [3]. To tackle these problems, many solutions exist like lexicon pruning, which enable to reduce lexicon size, or using n-grams of characters to deal with out of vocabulary words. However they hardly reach the state of the art performance. To the best of our knowledge, the largest lexicon used in the literature was composed of 200K words [4], and there could still occur out of vocabulary words (named entity, numbers, etc).

During the last years, significant progress in handwriting recognition have been made thanks to deep learning advances[5], namely with the Long Short Term Memories (LSTM) Recurrent Neural Networks (RNN) [6]. The LSTM recurrent neural networks achieve state of the art results in various applications involving sequence recognition, such as speech recognition [7], protein predictions [8], machine translation [9], and optical character recognition [10, 11]. The Connectionist Temporal Classification (CTC) [12] training framework allows frame level optimization while only providing the ground truth at word or line level. Therefore, it avoids a tedious manual segmentation of the im-

ages to provide the frame level ground truth. Performances of such networks in various application are due to their very high raw performances (i.e. without using additional linguistic resource) for character recognition [13]. These raw performances are not analyzed in the literature, but they provide hints that LSTM recurrent neural networks should be used for their properties and not only as a better character classifier as it is currently used in most of the works reported in the literature. In a recent study [14], it has been proved that a fully-connected deep network can reach a huge number of equivalent (but not equal) local minima during training, thus providing networks with overall similar performance while having internal dissimilar properties. These theoretical results have been the motivation for exploring new recognition and learning frameworks of LSTM neural networks. Breaking the standard use of LSTM RNN as a simple classifier introduced in a lexicon driven decoding scheme, this article explores new recognition and training strategies for LSTM RNN. The proposed new recognition paradigm improves handwriting recognition state of the art performance. Following the theoretical results in deep learning[14], we observe that multiple complementary LSTM networks can be obtained during the training stage. Instead of looking for the best network, as it is traditionally done, we obtain an ensemble of complementary networks which we call a cohort of LSTM networks. These complementary networks can efficiently be combined in a cascade[15]. Relying on the raw performances of each LSTM RNN, we show that recognition and rejection in the cascade can be implemented using a simple and fast lexicon verification strategy. This provides a new lexicon verification decoding paradigm for handwriting recognition, in opposition to the standard lexicon driven decoding scheme. This strategy reaches a very high precision regardless the size of the lexicon. As a consequence, the approach has no limitation regarding the lexicon size as demonstrated by the astounding results obtained using a gigantic lexicon of more than 3 million words.

This article starts by a review of the state of the art of handwriting recognition highlighting the latest results obtained with BLSTM networks and Hidden Markov Models. We also show the difficulty of using very large lexicons. In the

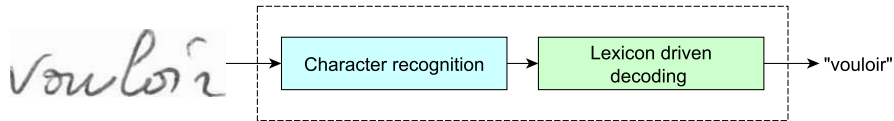


Figure 1: The standard handwriting recognition paradigm.

second part, we present both our lexicon verification strategy using a cascade of LSTM recurrent neural networks and the way a cohort of LSTM RNN can be obtained using specific training conditions. In the third part of the paper, the implementation of our method is described. Finally the results are presented on the Rimes and IAM databases, and then discussed before concluding.

2. Related works

2.1. Handwriting recognition

Handwriting recognition models can be classified according to the character segmentation approach which can be either explicit (an algorithm specifically segments characters prior to their recognition), or implicit (characters are classified without prior segmentation)[1]. It can also be classified according to the choice of the characters' recognition method (discriminant classifiers for hybrid approaches [16] or generative approach for Hidden Markov Models (HMM) [17]). The common point of these approaches is that they all rely on a lexicon driven decoding stage. Indeed, since character recognition is not perfect, the traditional word recognition strategy is to postpone the character decision process at the end of the sequence recognition process, where the best character sequence hypothesis being a valid sequence is finally selected. This traditional scheme is represented in figure 1.

In the literature, lexicon driven decoding methods are mainly based on Hidden Markov Models (HMM [18] and [19]). Their main strength is the segmentation free design, thus letting the decoding process provides the segmentation. Another interesting strength of HMM is their ability to easily integrate statistical language models (words n-grams), in addition to the definition of a lexicon

to work with. However using linguistic resources raises difficulties for lexicon driven decoding, which are discussed in the next paragraph.

2.2. The large vocabulary problem

When using lexicon driven decoding, the character classification decisions are postponed until the end of the word so as to decide for the most probable word belonging to the working lexicon. On one hand, a lexicon directed approach allows to correct character recognition errors. But on the other hand, it may produce errors by wrong corrections due to close words in the dictionary. The more words there are in the lexicon, the more likely an error can occur, and therefore the more difficult the recognition is. When using very large lexicons, such methods will generate more errors while requiring more processing time. In [10] the authors relate the results of the RIMES 2009 competition for isolated word recognition. Most of the participants have implemented HMM based approaches for which we can observe a neat difference of the recognition rate between small, medium and large lexicon size with a maximum size of 5 334 words. At this time, only the TUM system was implementing BLSTM RNN which was less sensitive to lexicon effects. The lexicon's size has also a major role in the use of these methods in real applications where the lexicon is linked to a language that can contain hundreds of thousands of words. In specific cases, using such linguistic resources to improve recognition is not anymore compatible with real time constraints.

The large vocabulary problem is well known, as described in [3], where a vocabulary of 1000 words is considered to be a large lexicon. Today larger lexicons are considered. In [13] the authors used lexicons with 50k, 12k and 95k words for respectively IAM, Rimes and OpenHaRT database, but none of this lexicon is fully covering the evaluation set. To the best of our knowledge, the largest lexicon ever used is composed of 200 000 words [4] using a n-gram model. But this is still generating problems as there is still out of vocabulary words. Moreover for real life application without any a priori knowledge of the vocabulary, lexicon for a language is up to hundreds of thousands words (French

Gutenberg dictionary has 336k words) to reach an acceptable coverage rate, but still without covering named entities, numbers, etc.

To overcome these limitations, solutions have been investigated: Pruning[20, 21, 22], lexicon free decoding[23, 24, 25, 26, 27], sub lexical units driven recognition[23, 28, 29].

Pruning methods exhibit other problems such as excessive pruning, leading to increasing errors.

Methods[23, 24] using hidden Markov models allow a lexicon free decoding , but they still have not reached the performance of the lexicon driven approaches. Note that they have also been used for numerical field recognition without lexicon [25, 26, 27], but digits are simpler to recognize than characters since they are generally isolated, and since there are only 10 classes.

Finally, in [23, 28, 29] the authors proposed a lexicon decomposition into prefix and suffix of the language, modeled by n-grams. These methods are based on statistics extracted from a training corpus. They exhibit state of the art performance when dealing with out of lexicon word [28].

As reviewed in this paragraph the large vocabulary problem is still an open question. With the current fast progress in deep learning, many architectures are studied. Regarding handwriting and speech recognition, it is currently lead by the Long Short Term Memory recurrent neural networks.

2.3. Recurrent Neural Networks

Recurrent neural Networks (RNN), proposed more than 30 years ago with Hopfield networks[30], get their efficiency from their ability to process sequences, thanks to recurrent connections bringing information about the previous inputs or states in the sequence. However, for a long period of time, training recurrent neural networks suffered from the vanishing gradient problem[31]. As a consequence long term dependencies couldn't be learned. Long Short Term Memory (LSTM) cells have thus been designed by Hochreiter et al. [6] in order to overcome this limitation. Many improvements still have been proposed on LSTM [32, 33].

LSTM cells enable to learn long or short term dependencies while processing sequences thanks to the introduction of gates (input, forget and output gates) followed by a sigmoid function, which control the memory (updating, resetting, expressing) by introducing a multiplier cell at each gate. RNN operates by processing the sequence in a particular direction, that is why bidirectionality has been introduced in RNN [34]. Then this idea has been extended to LSTM networks [7] to create bidirectional LSTM recurrent neural networks (BLSTM). However key applications like handwriting recognition are based on images which have two dimensions. In order to process images, multidimensional LSTM recurrent neural networks (MDLSTM) have been introduced [2].

LSTM networks only became popular once a new learning strategy has been introduced, the Connectionist Temporal Classification (CTC)[12], which is a neural variant of the well known Forward Backward algorithm used for training HMM. CTC greatly helps training such networks by allowing embedded training of character models from the word or sentence label without the need for knowing the location of each character. This is working especially well thanks to the introduction of a "joker" class between characters, which postpone the decision of the network until sufficient information is gathered along the sequence, so as to output the character hypothesis at one particular position in the input stream.

Today, the LSTM recurrent neural networks are the state of the art method for numerous sequence analysis applications, including optical character recognition [10, 35].

With such an architecture, LSTM networks provide nearly binary decision between probabilities for character and joker classes. One example of LSTM network outputs for the french word "demandeur" is given in figure 2, for readability of the legend, only lowercase characters are shown, and the correct characters are highlighted with colors.

Such an output profile allows to apply a simple decoding scheme without lexicon ("Best path decoding" in [36]) by taking the maximum class's posterior probability at each frame, and by removing every successive repetitions of each

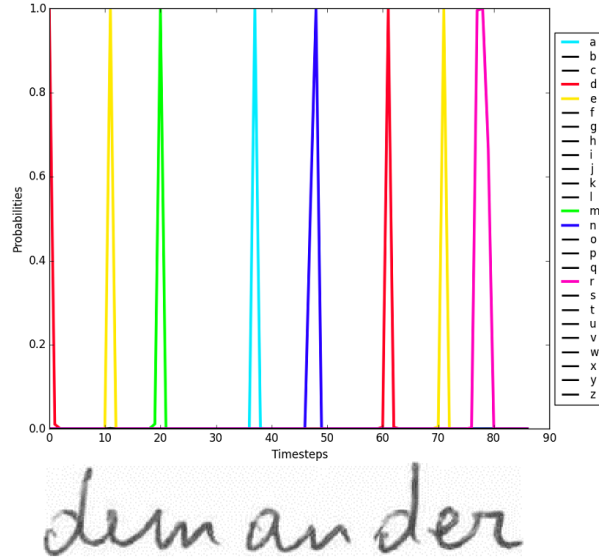


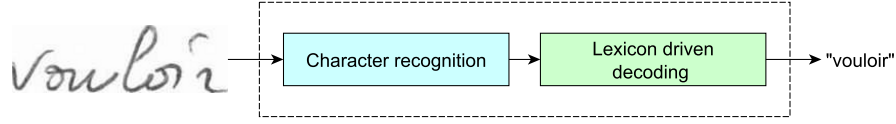
Figure 2: LSTM network outputs at the end of a CTC training. The outputs form peaks for every character recognized.

class (joker included), then the joker. Performances at the end of this lexicon free decoding scheme on a recognition task, although below state of the art, are very high. For example, on the Rimes word isolated recognition task, standard MDLSTM-RNN [13] recognize 67% of the words, by using the simple decoding strategy.

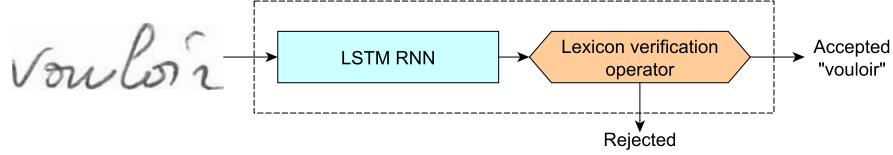
These LSTM RNN high raw performances demonstrate that in the traditional handwriting process, contribution to the recognition mainly comes from LSTM RNN or other neural networks architecture (see Fig. 3a). From these observations, we investigate how lexicon free decoding scheme could be improved by taking advantage of the raw performance of LSTM RNN.

3. Cohort of LSTM RNN combined in a cascade by verification

In this paper, we propose a new recognition paradigm (figure 3b) that take benefit from the high raw performance of LSTM networks in order to carry out a lexicon verification decoding strategy. This strategy is based on cascading



(a) The standard handwriting recognition paradigm



(b) Our new handwriting recognition paradigm

Figure 3: Handwriting recognition paradigm.

complementary neural networks, combined with a very reliable rejection stage based on lexicon verification.

3.1. The cascade framework

Cascading classifiers is a particular combination method, that combines classifiers decisions sequentially by exploiting the complementary behavior of the classifiers, in order to progressively refine recognition decisions along the cascade. The most famous contribution on cascade of classifiers is from Viola and Jones [15], who applied their popular framework to face detection. The authors present a cascade based on a large ensemble of diverse and weak classifiers, which enable a fast decision process by introducing sequential reliable decisions. The performance of each classifier may be low but it must exhibit a high confidence level. The decision stage is the most important part as it controls the overall performance.

In [37] and [38], an alternative cascade scheme is proposed by combining strong classifiers with different architecture and different input features. The strong classifier recognizes an important number of objects with a low error rate, while relying on a decision system allowing to transfer rejects to the next classifiers.

This literature review shows that the core of cascade of classifiers is the decision stage in charge of rejecting or accepting the recognition hypothesis. In the strong classifier approach, if an object is wrongfully accepted, then the cascade process ends. Whereas in the weak classifier hypothesis, it keeps going through the cascade and can eventually be rejected. Thus for cascade with strong classifiers, misclassified samples cannot be recovered, as the process stops once a classification is made. Ordering the classifiers in the cascade by decreasing reliability appears the best strategy. Improving the reliability of the decision stage can be achieved by combining multiple strong classifiers. One simple policy of combining classifiers is based on the agreement between the classifiers. We call the minimum number of agreement (MNA) the minimum number of classifiers that must take the same classification decision. This decision mechanism is essential to control the performance of the cascade while enabling to significantly speed up the process, when many classifiers are involved.

The second important element of a cascade is the classifiers complementarity. Indeed, almost similar classifiers would not allow to refine the decision along the cascade. Obtaining complementary classifiers can be done following two different approaches. The first one is by using different features sets such as in [38] The second one is by selecting the training sets, such as training on the rejects made by a previously used classifier. However designing many classifiers with different features is tedious and not straightforward, moreover the amount of training data decreases at each step of the cascade.

In this paper we propose an approach relying on the design of a cascade of LSTM RNN (figure 4) where complementarity of the classifiers is obtained by relaxing the learning rate of the RNN architecture and a reliable rejection stage is achieved through lexicon verification.

We first focus on the decision reliability of our lexicon verification operator, then we discuss how to generate complementary LSTM RNN.

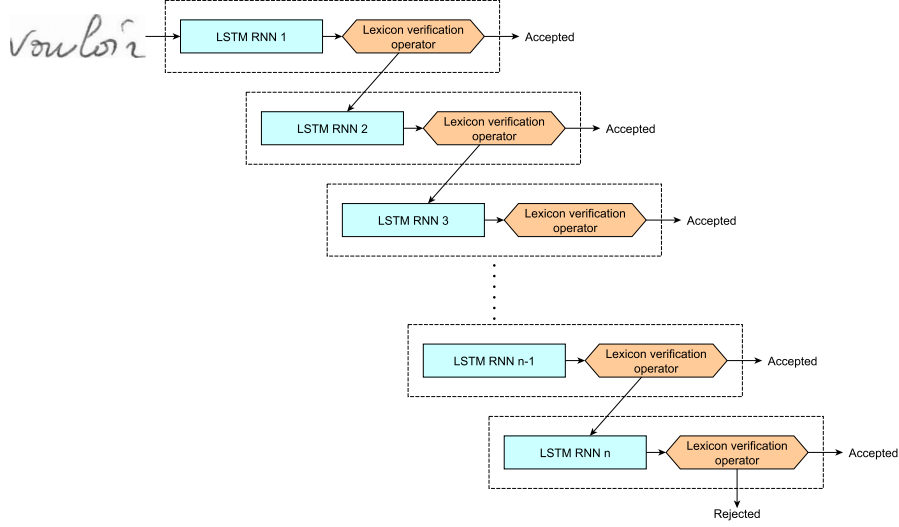


Figure 4: The proposed BLSTM Cascade. Each classifier process the rejects from the previous layer.

3.2. Lexicon verification reliability

Here, the objective is to validate or to reject any sequence of characters hypothesis produced by a lexicon free recognizer.

This verification process consists in accepting the characters string if it belongs to the lexicon, or rejecting it otherwise. Such simple verification stage can serve as a decision in the cascade only if it provides reliable decisions and has a very low false acceptance rate.

False acceptance (FA) decisions occur when the wrong character hypothesis produced by the recognizer matches one entry of the lexicon. Assuming independence between the lexicon free recognize and the lexicon, the probability of a false acceptance is simply the product $P_{wmis} \times P_{InLexicon} = P_{FA}$.

First the probability of making a mistake P_{wmis} is simply one minus the probability to have all characters recognized: $P_{wmis} = 1 - P_{wgood}$. And the probability of having all characters recognized is the probability of character recognition for one character to the power of the number of characters n in the word. The latest is simply one minus the Character Error Rate (CER) of the

classifier, giving: $P_{wmis} = 1 - (1 - CER)^n$. The probability of a word to be in the lexicon is the ratio of words of length n in the lexicon (m_n) over the number of all possible character string of length n . Assuming d characters classes we get $P_{InLexicon} = \frac{m_n}{d^n}$. Finally the probability of false acceptance of a character string is:

$$P_{FA} = (1 - (1 - CER)^n) \times \frac{m_n}{d^n} \quad (1)$$

Figure 5 shows the probability P_{FA} with respect to word length n for different CER, and for a lexicon composed of 336531 words from the french Gutenberg dictionary. The probability is very low for words of length 3 and more, and there is only a "high" probability for very short words of 1 or 2 characters. Notice that for a CER of 0.15, nearly above the BLSTM CER, the probability of a false acceptance is below 1%.

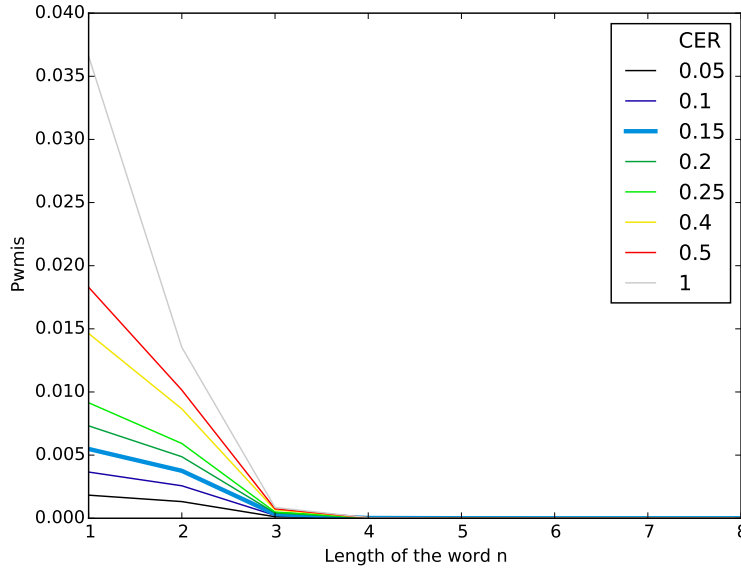


Figure 5: P_{FA} over the length of word n for different CER.

We now analyze if these theoretical considerations are verified in practice. In this respect, table 1 shows the performance obtained on the Rimes dataset using a lexicon free BLSTM recognizer. We see that a simple verification stage achieves a very low WER of only 2.25% while word recognition rate is very

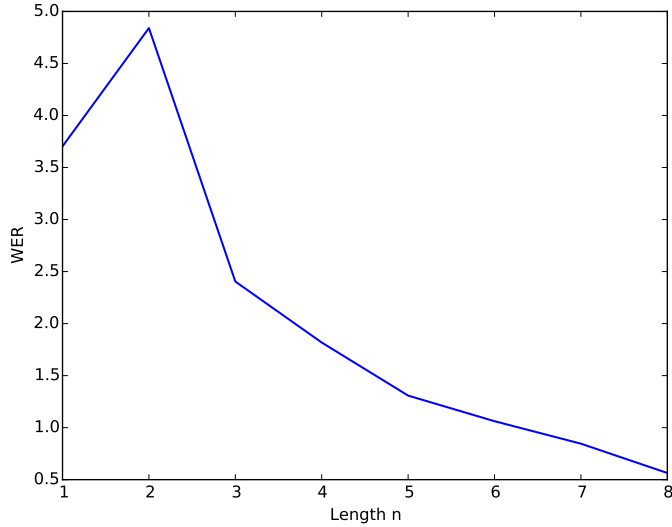


Figure 6: *WER* over the length of word n on the Rimes database.

high for a lexicon free recognizer. Most of the errors are type case, accents and plurals errors. Figure 6 shows the word error rate (or false acceptance rate) with respect to the word length obtained on the Rimes dataset. We see that highest WER rate occur for short words as predicted by our model. This first experiment demonstrates the strength of a verification strategy in combination with BLSTM classifiers.

Network	Recognized	Error	Rejection
BLSTM + checking	66.37	2.25	31.38

Table 1: Results of our lexicon checking strategy applied to a BLSTM network on the Rimes database.

Lexicon Verification is a rejection rule with a very low average error that can serve as a rejection stage in a cascade of word recognizer. Now we investigate our second problem, how to generate complementary LSTM RNN.

3.3. Generation of complementary LSTM RNN

There are many ways to generate complementary LSTM neural networks. The easiest one is by exploring different architectures (BLSTM vs MDLSTM, or changing the number of layers or neurons, etc.) or exploring different input features (pixels, Histogram of Gradients, etc.). However these modifications are costly in terms of design, training time, and limited in number.

Some previous experiments have shown that similar LSTM recurrent neural networks trained with different initial weights, can be combined with success [11]. These networks have similar recognition rates but the connection weights are different. However training many networks starting with different initializations is long as it takes up to a week to train a single network.

In order to get as much complementary networks as possible with a limited effort considering both time in design and training, the idea lies in controlling the training phase of deep neural networks, and is inspired by the work of Choromanska et al in [14]. This work makes a parallel between fully-connected neural networks loss function and high degree random polynomials which have a huge number of local minima with equivalent magnitude. The authors conclude that when training a large neural network, reaching a local minimum is nearly similar to reaching the global optimum. As a consequence, exploring these local minima may be a simple but relevant strategy to obtain a large amount of complementary networks that perform equally well, but with different local properties. We call the ensemble of networks obtained by this strategy a cohort, and we use the obtained cohort to feed a cascade.

Nevertheless, this easy and fast strategy to get many complementary networks requires some attention on the training parameters in order to get the desired property. Indeed training must avoid to be trapped in one local minimum in order to get complementary classifiers that will correspond to many different local minimum. Training a neural network using steepest gradient descent is controlled by three important parameters which are: the learning rate, the momentum and shuffling the data set. Momentum as is the factor that control the weights update and thus the convergence of the training procedure. We

use a fix momentum of 0.9, which is commonly used in the literature, in order to help escaping local minimum. When training networks, the examples are shuffled between each epoch in order to get a better convergence by preventing cycles. As training examples are randomly shuffled between epochs, we can consider that each epoch is randomly reached, and by extension that the weights of the network "randomly" reach their values. By doing so the network extracted at each epoch of one single complete training phase using shuffling and momentum seems to fulfill the requirements to get complementary networks. The last and maybe most important parameter is the learning rate. A large learning rate will not result in a good convergence of the network and a too small one will lead to very small changes in weights values and few changes of local minimum, and as a consequence gives no complementary networks between epochs.

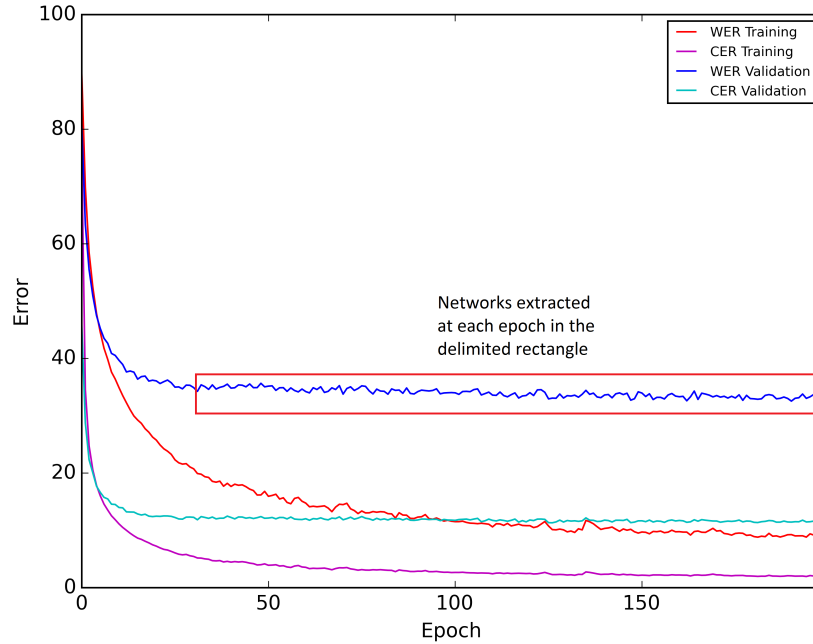


Figure 7: Learning curves of a network with learning rate fixed at 10^{-4} . There is local fluctuations on the error.

Figure 7 shows that the WER and the CER both fluctuate, ensuring the

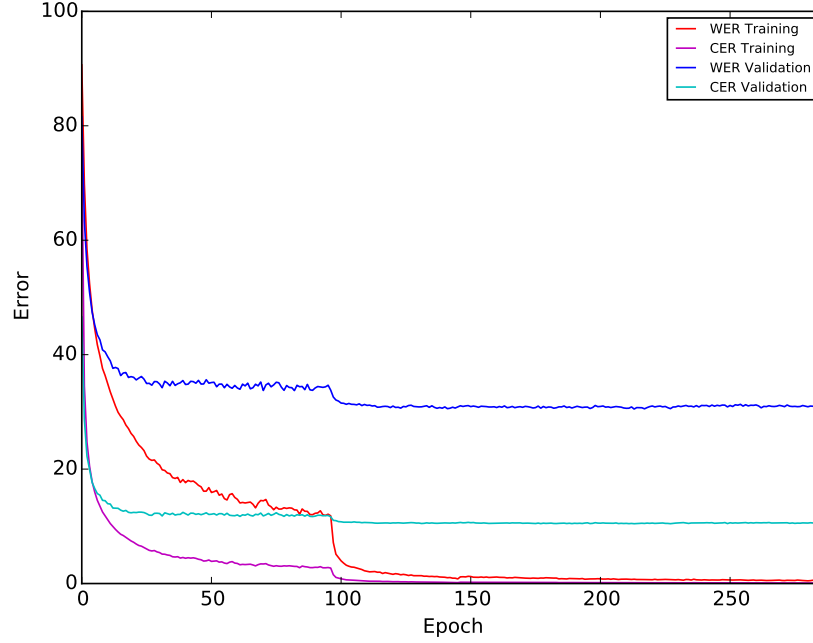


Figure 8: Learning curves of a network with learning rate at 10^{-4} then 10^{-5} at epoch 96. The error fluctuates less after the learning rate has decreased.

variability between the networks and thus their complementarity. This behavior is obtained for a fixed learning rate (10^{-4}). The red rectangle on figure 7 shows the region where the networks can be selected by our strategy at each epoch, since the WER is at a stage of equivalent performance and does not decrease significantly on the validation set. A larger learning rate would produce more fluctuations, and then probably more variability. However as shown on figure 9 the network does not converge properly with a learning rate of 10^{-3} , as a consequence it gets a worst CER and WER. Decreasing the learning rate to 10^{-5} at the epoch 96 (as shown on figure 8) implies less fluctuation on the WER leading to better performances but less variability and then nearly no complementarity.

For this task a learning rate of 10^{-4} is optimal. This was confirmed by analyzing 100 networks taken during the epochs of the training phase shown

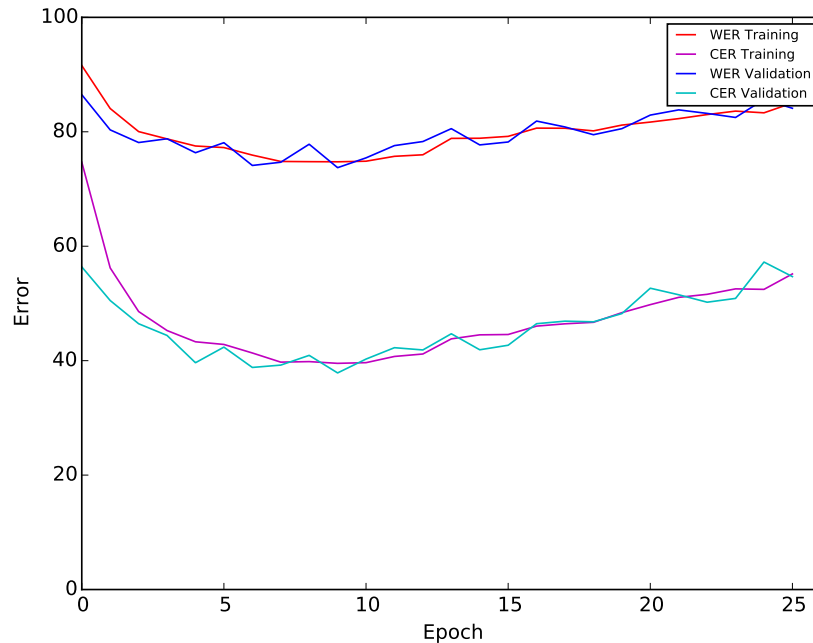


Figure 9: Learning curves of a network with learning rate fixed at 10^{-3} . Training has not converged.

on 7. They exhibit dissimilar local properties measured as the percentage of identical answers between two networks. We measured a average similarity of 66.7% between two networks with a standard deviation of only 0.009, proving the complementarity of our networks (whereas for a learning rate of 10^{-5} the mean similarity is 90.4%). The proposed training strategy although simple is very effective to get complementary networks.

At this stage, we propose a new paradigm for handwriting recognition that is based on cascading LSTM RNN classifiers. In this section we have demonstrated that lexicon verification can serve as an effective reliable rejection rule in the cascade, while training complementary LSTM RNN can be achieved by choosing some modified hyper parameters.

4. Implementation and Results

Based on the methodological principles established in the previous section, we now detail our implementation and present our results achieving state of the art performances.

4.1. Dataset and Evaluation

The experiments have been carried out on the Rimes[10] and IAM [39] isolated words datasets. The provided split and lexicon of each database is used. The char sets contain Latin alphabet upper and lower case, symbols that may occur in a word like "''" or "-", and many others. For the Rimes database there is also accented characters, and the evaluation is made on lower case as in [11] for comparison purpose and also because of ground truth ambiguity on certain upper case characters especially for the letter "j" in the word "je" or "j" where many ground truth errors occur in the dataset.

The lexicons for both dataset are composed of all the words of the training, validation and test sets as used in the competitions and related papers, giving lexicons size of 5744 and 12202 for Rimes and IAM respectively. A smaller lexicon composed of test words (1692) for Rimes is used in order to evaluate lexicon sensitivity.

In order to evaluate the performance of our approach on very large lexicons, we also used two extremely large lexicons for each dataset, collected by ourselves:

- **3 276 994** words from the french Wikipedia, Wiktionnaire, the French dictionary Gutenberg and the Rimes dataset ;
- **2 439 432** words from the one billion word data [40] and the IAM dataset.

For both datasets, we measure the performances with the Character Error Rate (CER), calculated with the Levenshtein distance, and the Word Error Rate (WER). We also measure the Word Recognition Rate WRR and the Word Rejection Rate WJR. In the classifier cascade alone character errors on the rejected words are not considered in the CER calculation.

4.2. Architecture

Two well established LSTM RNN architectures have been used for the experimentations:

The first is a BLSTM architecture identical to the one used in [41]: it is a two layers network composed respectively of 70 and 120 LSTM blocks, separated by a subsampling layer of 100 hidden neurons without bias, and with an hyperbolic tangent activation function. The network also has two layers to reduce the sequence length. The first one concatenates the input vectors in pairs, while the second one concatenates the output vectors of the first layer in pairs. For example, a sequence composed of 12 frames of 1 pixel width is transformed into a sequence of length 3 corresponding to 3 groups of 4 pixels width. As input features, we use histogram of oriented gradients (HOG) [42] that have demonstrated their efficiency for handwriting recognition [43]. Images are normalized to 64 pixels height. A sliding window of 8 pixel width extracts the HOG features at a 1 pixel pace. This architecture has been selected for its balanced characteristics, both performing slightly better than the reference architecture [2] and allowing a fast decoding, 15 milliseconds per word on average.

The second is a MDLSTM architecture similar to the reference architecture in [2]. This architecture is composed of three layers of respectively 2, 10 and 50 LSTM cells, and two subsampling layers of 6 and 20 neurons. Raw pixels of the image are normalized and directly fed to the network. The decoding time is 10 milliseconds per word on average.

For both architectures we use a lexicon free "Best path decoding" algorithm [36] described in (2.3) to retrieve the characters string. Training and decoding have been performed with RNNLIB [44].

4.3. Generation of a cohort of LSTM RNN

To get a cohort of complementary LSTM RNN we resort to using the three following strategies: (i) Our training trick, for which we get one network per epoch ; (ii) Different starting initialization ; (iii) Two different architectures BLSTM and MDLSTM.

Moreover we use a common trick to improve performances of neural networks, we use transformations to increase the size of the training set. By using rotations and warping, we multiply the training set by 3, and as shown in figure 10 the sequence and label error improve. Beside providing a lower error, the transformations also bring more fluctuations during training, thus improving the complementarity.

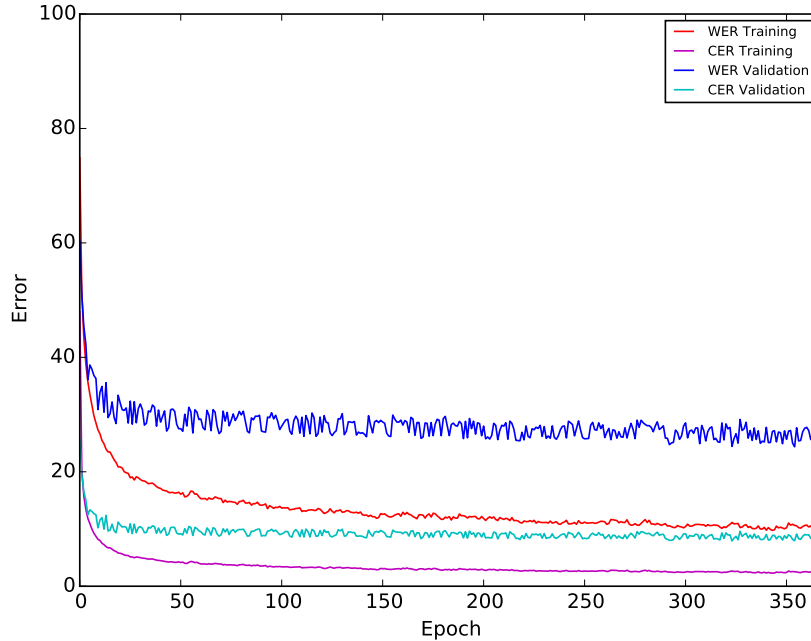


Figure 10: Learning curves of a network with transformations with learn rate fixed at 10^{-4} . The error fluctuates at a higher amplitude, SeqError is the WER and LabelError the CER.

2100 different networks are collected from only ten different trainings on the Rimes database with different random initializations:

- 2 BLSTM trainings;
- 1 MDLSTM training;
- 4 BLSTM trainings with transformations;

- 3 MDLSTM trainings with transformations.

In order to prove that our method isn't database dependent, we also trained two networks with transformations, two BLSTM and two MDLSTM, on the IAM database, giving a total of 1039 networks for this task.

The purpose of doing multiple training is multiple, it allows us to: i) get more networks with training run in parallel, (ii) check successfully that the method doesn't depend on the initial weights or architecture selected, and (iii) improve performances thanks to the two architectures using different features.

4.4. Results

4.4.1. Cascade

Remember that tuning a cascade depends on ordering the classifiers in the cascade, and on the Minimum Number of Agreements (MNA) in the decision stage. Ordering the classifier in the cascade is made according to the deletion error criterion on the validation set. Tuning the MNA parameter was carried out by considering long words and short words.

As previously seen in figure 5, we observe that words with three characters and less are more prone to mistakes, justifying this choice. In these experiments we have chosen a MNA of 3 for long words and 10 for short words, however, small deviation from the values do not affects the results much. For example when using MNA in the range 2 to 10 for long words, and MNA in the range 5 to 40, we observe no modification in the CER, while WER is modified by 0.25% only. It is also possible to optimize this parameters regarding the validation set.

First we report our results on the Rimes dataset in table 2, we achieve state of the art performance for the 5k words lexicon. We compare our results to the state of the art on the Rimes dataset with the large lexicon. To compare our method with others, which have no reject, we use a Viterbi lexicon decoding on the rejected words at the end of the cascade. For doing so, character posterior probabilities are the average class posterior probabilities of ten MDLSTM randomly picked among the cohort, before running the Viterbi decoder. Averaging

the output probabilities from different networks improves performances (comparing to taking the output of only one network). Notice that the number of networks randomly selected has a very low impact on the performance above a certain threshold. Selecting 5, 10, 20 or 40 networks yields nearly similar results. The gap between this study and the results presented in [29], both in terms of CER and WER is significant with an absolute decrease of 0.56 (29%) of the CER and 0.42 (11%) of the WER.

The recognition difference between small and large lexicons is very small, there is only 0.48 points in difference. The difference between the large and gigantic lexicon is more important, however it is still low with a difference of 5.71%. Both results prove the very low sensitivity of the method to the lexicon size.

Regarding the gigantic lexicon, the recognition rate is very high 90.14% as the lexicon is nearly 600 times larger than the Rimes lexicon. Notice that we have no comparison with any other study as there is no other reference in the literature using such gigantic lexicon size.

System	Lexicon size	WRR	WER	WJR	CER
This work	<i>1692</i>	96.08	2.32	1.61	0.76
This work	<i>5744</i>	95.60	3.00	1.40	0.99
This work	<i>3 276 994</i>	90.14	9.18	0.68	2.67
This work + Viterbi	<i>5744</i>	96.52	3.48	-	1.34
Menasri et al. [11]	<i>5744</i>	95.25	4.75	-	-
Poznanski et al. [29]	<i>5744</i>	96.10	3.90	-	1.90

Table 2: Results of the 2100 networks cascade on the Rimes database.

Results on the IAM dataset are presented in table 3. We obtain a significantly lower WER without Viterbi decoding of the rejects, and nearly the same WER when adding Viterbi decoding of the rejects. The CER is below state of the art by 0.45 (13%). The result for the gigantic lexicon is also impressive with 85% of words recognized with a lexicon 200 times larger.

System	Lexicon size	WRR	WER	WJR	CER
This work	12202	91.20	5.35	3.62	2.13
This work	2 439 432	85.06	13.30	1.64	4.68
This work + Viterbi	12202	93.45	6.55	-	2.99
Poznanski et al. [29]	12202	93.55	6.45	-	3.44

Table 3: Results of the 1039 networks cascade on the IAM database.

For both dataset our approach achieves state of the art results, and it has a low sensitivity regarding the lexicon size and can deal with gigantic lexicons.

4.5. Processing time

One drawback of our approach is that it requires many networks and thus a large processing time. However the cascade architecture is indeed very effective regarding computation cost, because once a candidate is accepted by the verification stage it does not pass through the rest of the cascade. The processing time to do the lexicon verification is below one microsecond even with gigantic lexicons. When looking at our system with 2100 networks, the mean processing time per word is 729 milliseconds, furthermore 80% of the words are processed in less than 0.175s (after 14 networks or less) and 90% in less than half a second (after 41 networks or less). This timing has been evaluated on an intel cpu i7-3740QM. Also networks as light as the one we used doesn't take a lot of memory space, as the whole 2100 networks fit into 6GB when coding them with double precision (64 bits).

Above timing consideration, one would ask whether the whole cohort of networks is necessary, or if a selection of them would perform similarly, which in this case would also lower the processing time. This point is addressed in the next section.

4.6. Selection of complementary classifiers

The aim of the selection is to decimate the cohort of classifiers so as to keep the most efficient reduced set of classifiers. To do so we removed all networks which have poor performance in the cascade on the whole validation set. Networks which don't recognize new words or which make too much false acceptance are removed. By doing so we manage to reduce the number of networks from 2100 to 118. We observe a

performance loss when compared to the cohort of 2100 networks as shown in table 4. However, nearly similar results are obtained when using Viterbi decoding of the rejects, improving state of the art with this reduced architecture also: the WER is **3.64%** (previously 3.48%) and the CER is **1.49%** (previously 1.34%).

System	Lexicon size	WRR	WER	WJR	CER
This work	<i>5744</i>	92.96	2.28	4.76	0.69
This work	<i>3 276 994</i>	88.82	8.04	3.14	2.20
This work + Viterbi	<i>5744</i>	96.36	3.64	-	1.49

Table 4: Results of the pruned cascade of 118 networks on the Rimes database.

5. Conclusion

This work presents a new paradigm, the combination in a cascade of a cohort of LSTM recurrent neural networks, and explains how to obtain it and how to combine it. We successfully achieve state of the art results for both Rimes and IAM datasets, with no parameters to tune, even if you can customize the system for application’s needs. The lexicon verification operator was key to the success of the method and the very low sensitivity to lexicon size. On top of that, our method allow to process gigantic lexicon, with a low sensitivity to the size of the lexicon, which has never been done before. This work is applied to isolated word recognition, however most of the application are at the line level, one interesting perspective is to extend this idea at a line level.

The way we obtain our cohort of networks and the results we get raises some interrogations, especially about the current trend of deeper and deeper networks or more complex architectures. Is covering a huge number of local minimum with simple networks and combined their results in simple ways working for every application ? Does it works with every kind of network, like convolutional neural networks ? Extending our cohort principle to other neural networks and applications, like image description seems to be a promising path to explore.

References

References

- [1] R. Plamondon, S. N. Srihari, Online and off-line handwriting recognition: a comprehensive survey, *IEEE PAMI* 22 (1) (2000) 63–84.
- [2] A. Graves, J. Schmidhuber, Offline handwriting recognition with multidimensional recurrent neural networks, in: *NIPS*, 2009, pp. 545–552.
- [3] A. L. Koerich, R. Sabourin, C. Y. Suen, Large vocabulary off-line handwriting recognition: A survey, *Pattern Analysis & Applications* 6 (2) (2003) 97–121.
- [4] M. Hamdani, P. Doetsch, M. Kozielski, A. El-D. Mousa, H. Ney, The rwth large vocabulary arabic handwriting recognition system, in: *IAPR International Workshop on Document Analysis Systems*, 2014, pp. 111–115.
- [5] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *Nature* 521 (7553) (2015) 436–444.
- [6] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural computation* 9 (8) (1997) 1735–1780.
- [7] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, *Neural Networks* 18 (5) (2005) 602–610.
- [8] T. Thireou, M. Reczko, Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins, *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 4 (3) (2007) 441–446.
- [9] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: *NIPS*, 2014, pp. 3104–3112.
- [10] E. Grosicki, H. El Abed, Icdar 2009 handwriting recognition competition, in: *ICDAR*, 2009, pp. 1398–1402.
- [11] F. Menasri, J. Louradour, A.-L. Bianne-Bernard, C. Kermorvant, The a2ia french handwriting recognition system at the rimes-icdar2011 competition, in: *Document Recognition and Retrieval XIX*, 2012, pp. 82970Y–82970Y.

- [12] A. Graves, S. Fernández, F. J. Gomez, J. Schmidhuber, Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, in: ICML, 2006, pp. 369–376.
- [13] V. Pham, T. Bluche, C. Kermorvant, J. Louradour, Dropout improves recurrent neural networks for handwriting recognition, in: International Conference on Frontiers in Handwriting Recognition, 2014, pp. 285–290.
- [14] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, Y. LeCun, The loss surfaces of multilayer networks., in: AISTATS, 2015.
- [15] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: CVPR, Vol. 1, IEEE, 2001, pp. I–511.
- [16] A. Senior, T. Robinson, Forward-backward retraining of recurrent neural networks., NIPS (1996) 743–749.
- [17] A. El-Yacoubi, M. Gilloux, R. Sabourin, C. Y. Suen, An hmm-based approach for off-line unconstrained handwritten word modeling and recognition, IEEE PAMI 21 (8) (1999) 752–760.
- [18] L. R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proceedings of the IEEE 77 (2) (1989) 257–286.
- [19] T. Plötz, G. A. Fink, Markov models for offline handwriting recognition: a survey, International Journal on Document Analysis and Recognition 12 (4) (2009) 269–298.
- [20] S. Madhvanath, V. Govindaraju, Holistic lexicon reduction for handwritten word recognition, in: Document Recognition III, 1996, pp. 224–234.
- [21] S. Madhvanath, V. Krpasundar, V. Govindaraju, Syntactic methodology of pruning large lexicons in cursive script recognition, Pattern Recognition 34 (1) (2001) 37–46.
- [22] M. Gilloux, Réduction dynamique du lexique par la méthode tabou, in: Colloque International Francophone sur l’écrit et le Document, 1998, pp. 24–31.

- [23] A. Brakensiek, J. Rottland, G. Rigoll, Handwritten address recognition with open vocabulary using character n-grams, in: International Workshop on Frontiers in Handwriting Recognition, 2002, pp. 357–362.
- [24] A. Bharath, S. Madhvanath, Hmm-based lexicon-driven and lexicon-free word recognition for online handwritten indic scripts, IEEE PAMI 34 (4) (2012) 670–682.
- [25] M. Shridhar, G. Houle, F. Kimura, Handwritten word recognition using lexicon free and lexicon directed word recognition algorithms, in: ICDAR, Vol. 2, 1997, pp. 861–865.
- [26] C. Chatelain, L. Heutte, T. Paquet, A two-stage outlier rejection strategy for numerical field extraction in handwritten documents, in: ICPR, Vol. 3, 2006, pp. 224–227.
- [27] C. Chatelain, L. Heutte, T. Paquet, Segmentation-driven recognition applied to numerical field extraction from handwritten incoming mail documents, in: Document Analysis System, LNCS 3872, Springer, 2006, pp. 564–575.
- [28] M. Hamdani, A. E.-D. Mousa, H. Ney, Open vocabulary arabic handwriting recognition using morphological decomposition, in: ICDAR, IEEE, 2013, pp. 280–284.
- [29] A. Poznanski, L. Wolf, Cnn-n-gram for handwriting word recognition, in: CVPR, 2016, pp. 2305–2314.
- [30] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the national academy of sciences 79 (8) (1982) 2554–2558.
- [31] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.
- [32] F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm, Neural computation 12 (10) (2000) 2451–2471.
- [33] F. A. Gers, N. N. Schraudolph, J. Schmidhuber, Learning precise timing with lstm recurrent networks, Journal of Machine Learning Research 3 (2003) 115–143.

- [34] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, *Signal Processing, IEEE Transactions on* 45 (11) (1997) 2673–2681.
- [35] H. El Abed, V. Margner, M. Kherallah, A. M. Alimi, Icdar 2009 online arabic handwriting recognition competition, in: *ICDAR, 2009*, pp. 1388–1392.
- [36] A. Graves, *Supervised sequence labelling with recurrent neural networks*, Vol. 385, Springer, 2012.
- [37] B. Zhang, Reliable classification of vehicle types based on cascade classifier ensembles, *Intelligent Transportation Systems* 14 (1) (2013) 322–332.
- [38] P. Zhang, T. D. Bui, C. Y. Suen, A novel cascade ensemble classifier system with a high recognition performance on handwritten digits, *Pattern Recognition* 40 (12) (2007) 3415–3429.
- [39] U.-V. Marti, H. Bunke, The iam-database: an english sentence database for off-line handwriting recognition, *International Journal on Document Analysis and Recognition* 5 (1) (2002) 39–46.
- [40] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, T. Robinson, One billion word benchmark for measuring progress in statistical language modeling, *arXiv preprint arXiv:1312.3005*.
- [41] L. Mioulet, G. Bideault, C. Chatelain, T. Paquet, S. Brunessaux, Exploring multiple feature combination strategies with a recurrent neural network architecture for off-line handwriting recognition, in: *Document Recognition and Retrieval, 2015*, pp. 94020F–94020F.
- [42] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: *CVPR, Vol. 1, IEEE, 2005*, pp. 886–893.
- [43] G. Bideault, L. Mioulet, C. Chatelain, T. Paquet, Spotting handwritten words and regex using a two stage blstm-hmm architecture, in: *Document Recognition and Retrieval, 2015*.
- [44] A. Graves, Rnnlib: A recurrent neural network library for sequence learning problems, <https://sourceforge.net/projects/rnnl>.